Title	Freeware, Shareware, and Open-Source Mathematical Software Tools: a Feasible Alternative to Commercial Symbolic Packages for Mathematics Education (Mathematical Software and Education : Study on effective use of Mathematical Software)
Author(s)	Iglesias, Andres
Citation	数理解析研究所講究録 (2013), 1865: 204-214
Issue Date	2013-11
URL	http://hdl.handle.net/2433/195361
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
第1865巻 2013年 204-214

Freeware, Shareware, and Open-Source Mathematical Software Tools: a Feasible Alternative to Commercial Symbolic Packages for Mathematics Education

Andrés Iglesias[1,2]
[1]*Department of Applied Mathematics and Comp. Sciences*
University of Cantabria, Avda. de los Castros
s/n, E-39005, Santander, Spain
[2]*Department of Information Science*
Faculty of Sciences, Toho University
2-2-1 Miyama, 274-8510, Funabashi, Japan
E-mail: iglesias@unican.es
Web site: http://personales.unican.es/iglesias

(This paper is specially devoted to my longtime dear friend Professor Setsuo Takato, who is now writing beautiful lines in a new chapter of his book of life. Best wishes for this new stage of his life to be also brilliant, exciting and successful. Thanks for allowing me to become part of your life. Doomo arigatoo!!!)

Abstract

This paper corresponds to the printed form of an invited talk the author delivered in a RIMS workshop held at Kyoto University in August 2012. The main core of the presentation was about the new challenges we face as educators of Mathematics subjects at University level and how to solve them. To this aim, the talk proposed an intensive use of freeware (mostly symbolic) computational tools, as a way to overcome both instructional and economical issues. During the talk, it was emphasized that current freeware symbolic tools are powerful enough to represent a feasible, reliable alternative to costly commercial symbolic packages. The talk also explores the reasons behind the blossoming of this new freeware solutions for mathematical computation.

1 Introduction

On August 22nd. 2012, the author was kindly invited to deliver a talk at the *"RIMS Workshop on Mathematical Software and Education: Study on Effective Use of Mathematical Software"*, held at the Research Institute for Mathematical Sciences (RIMS) of

Kyoto University, Kyoto, Japan (`http://www.kurims.kyoto-u.ac.jp/en/index.html`). The primary goal of this workshop was to analyze the interplay between mathematical software (mostly CAS, computer algebra systems) and education regarding the effective use of mathematical software in order to improve the educational level and bring out the best of the students at all educational steps.

We are indeed facing a new era in which our students have to meet the new challenges of an increasingly technological world. In this regard, it is clear that old approaches to education are no longer applicable and hence, new educational paradigms must be envisioned and implemented. This challenge is specially important in very difficult subjects such as Mathematics and other scientific disciplines, which typically suffer from high failure rates. Several academic reports have pointed out the difficulties our students face when studying (and suffering) mathematical subjects. Students and professors at university often cite lack of preparation from high school, poor study habits and the rapid pace of the course as reasons for such low scores, but it is clear that there are many reasons to explain why this problem arises recurrently everywhere.

Another issue is to recognize the profile of our current students, as they are quite different to those of previous decades. In general, they are less skilled than their counterparts in the last decades in deduction, mathematical intuition and scientific reasoning and encounter more problems in solving questions with scientific content. Their background is also less solid in both science and arts. Furthermore, they also have less oral and written communication skills, with a much limited vocabulary and hence find some troubles for a full comprehension of concepts and ideas. Very often, they lack discipline and exhibit poor study habits such as poor note-taking skills, poor time management, last minute work, procrastination, over-reliance on classmates and/or Internet and so on. On the positive side, most current students come to college and university with greater computer proficiency and technology skills than their predecessors. Technology is natural to them as they got accustomed to use it from their childhood. Today's students' equipment is by far the most complete and varied we have ever seen: modern connectivity devices such as last generation smartphones, powerful laptops, USB memory cards, webcam, MP3 player, memory sticks, digital camera and smart cards. Very often also Internet connection at home, a desktop computer, videogame consoles, wide flat screens, videotape players and recorders, cable and/or satellite TV, and videocamera. Some students also have GPS, beam proyector, Blue-ray player/recorder, car navigator and other sophisticated electronic devices. They are familiar with terms such as pixel, texturing, RGB color palette, and technologies such as remote control, Internet surfing, DVI and HDMI connectors and many more. Much better, they are not only accustomed to technology but also they know how to use it efficiently. Therefore, proper use of computer tools and other technology turns out to be more than appropriate to promote their background to an upper level [2, 3].

1.1 The role of educational materials

In our opinion, part of the solution must come from a good collection of supporting materials, of which computer software is undoubtely a primary resource. Mathematics is very much practice-based. Students may grasp a concept in the classroom, but they will certainly lose it if not reinforced by homework. For doing so, students need to be

provided with good supporting materials so that they can effectively learn by themselves. High-quality, helpful educational materials allow underachieving students catching up on belated assignments and get extra time for successful backtracking. It is at this point where Computer Algebra Systems (CAS onwards) can really pave the way, making the most with less. In author's opinion, CAS are very powerful tools (arguably the best ones) to face the challenges of this new approach to Higher Education.

Fortunately, there is a wealth of computational software coming in our help. Among them, the symbolic computational programs have already proved to be very effective tools in order to improve the general performance of our students both at the classroom and for homework. There is a limiting factor, however, in this educational approach. The most popular symbolic tools so far are commercial software and, in general, tend to be costly for standard students. In fact, we have witnessed a big rise in the cost of this symbolic software. Even although the companies developing this kind of software have tried to create special client license programs in order to increase their customer base and allow more people to access their programs, it is clear that economical issues have become increasingly important.

As a consequence, both individual users and academic institutions are nowadays struggling to satisfy the increasing rates of prices for commercial mathematical software. In addition, the companies have generally fail to involve users in the process of updating their products. It is a typical complaint from many users that the new versions of very popular commercial programs are sometimes quite different than previous versions, very often involving substantial changes in the syntax and structures of the programming language, core of commands, graphical capabilities, and the like. This clearly affects users' performance by increasing the time to get familiar with the new version and, therefore, lowering their learning curve. While it is clear that the primary goal is to improve the product and provide users with a better *"user experience"*, it often happens that economical motivations are also part of the equation. In a contradictory movement, some companies have established the tradition to launch versions of their software according to a prescribed schedule, even if they have little (or nothing) of interest to offer at that time. Both undesirable situations may coexist in such a way that some versions are completely irrelevant while others are too different. A classical example of the former is given by the Premium licenses that usually provide access to the new updates of the product for either a limited number of versions or a limited time. The disappointment comes when you discover that the new version provided by the Premium service license is basically the same you bought at the time of applying for that service, with no relevant features at all. By Murphy's law, you have probably to wait until the new version (not covered by the Premium service) to really find a major update of the program.

As a summary, users of commercial mathematical software are facing great challenges in both computational and economical terms. The former is hard (if not impossible) to be solved, since it relies completely on the software developers side. Typically, software companies are reluctant to give up part of their business to their customers and prefer to have full control of their products. While many software developers hear their customers and try to incorporate their suggestions into the new versions of their products, users play a very limited role (if any) into the strategic decisions of the company. The economical issue has also been a very limiting factor for years. Mathematical software can be very expensive. One can argue that these systems are usually very powerful, easy to use,

are equipped with a nice graphical user interface and good graphical capabilities, are very flexible and come with a nice documentation and support. But still, they are very expensive, that's the bottom line.

A new actor in this scenario is the appearance and growth of a new generation of powerful freeware mathematical programs able to compete on an equal basis with commercial solutions. The fact that they are provided for free does not mean they are less powerful than their commercial counterparts. Besides, they are not subjected to the typical constraints that proprietary systems have; depending on the kind of license they have, these programs can usually be freely downloaded, installed, modified and/or distributed. Furthermore, most of these programs are created by communities of users, thus allowing end users to participate actively in the new versions of such programs. This open architecture has many obvious advantages for users far beyond the economical considerations. In most cases, the source code can be freely accessed and modified, so that users can effectively adapt the software to their particular needs. In other cases, a powerful programming language is provided, so that individual users can create their own functions and libraries in a transparent way. This means that these new functions and libraries are treated by the system as native libraries, and then, they can be invoked basically at will provided that they respect the programming guidelines and syntax of the underlying programming language.

In this paper we explore some aspects of this on-going process currently happening within the mathematical community as a results of this shift from commercial to freeware mathematical software. In particular, next section discusses some issues related to this new approach, such as the kind of licenses available and their implications for further use. We also provide pointers to some open-source and freeware tools and middleware available to handle a variety of tasks regarding mathematical education.

2 Free/Share/Open-Source Software

2.1 Clarifying the terms

A very important issue is to distinguish clearly among what is freeware, free software, open-source software and shareware. Although there is a common misconception portrayed in many web sites and media about these terms being synonymous, they actually refer to qualitatively different legal scenarios. The importance of this question for end-users and companies becomes clear as soon as you want to distribute your digital creations: being unaware about which kind of legal situation the software you use is might potentially lead to legal liability for using such software without the required permissions (licenses). To make things tougher, there is not an unanimous opinion about the right meaning of those terms and their legal implications, so users should always read the corresponding license agreements as they are the only reliable source when it comes to legal rights and obligations. As such, the description of terms below is mostly based on commonly accepted practices within relevant software communities rather than well-established "official" definitions themselves.

By *freeware* we refer to computer software available for use at no cost or for an optional fee. Typically, freeware is fully functional and offered for an unlimited time, although some restrictions might apply. Of course, this concept can also include proprietary soft-

ware provided that it is offered for free. In this sense, freeware is used to refer to software which is simply free-of-charge, with no other implications behind. Therefore, it must not be confused with *free software*, the later meaning that it can be used, studied, and modified without restriction. Although proprietary software companies typically use the term "free software" to refer to price, free software is actually a matter of freedom, not price. In words of Richard Stallman in the famous "The Free Software Definition" document [8] released by the Free Software Foundation (FSF), an organization that advocates for free software [7], to understand the concept, one should *"think of free as in free speech, not as in free beer"*.

By *open-source software* we mean computer software that is available in source code form. Usually, such source code and certain other rights normally reserved for copyright holders are provided under a software license that allows users to study, change, and improve the software. According to this concept, freeware may or may not be open-source, depending on whether or not the source code is also provided. Similarly, free software may or may not be open-source software, depending on the rights provided regarding the re-use of such software and subsequent distributions. This is, however, a matter of discussion. For the Free Software Foundation, free software is computer software liberally licensed to grant the right of users to use, study, change, and improve its design through the availability of its source code. From this standpoint, all licenses qualified as free software are also considered open-source licenses, but this criterion is not unanimously acknowledged.

Open-source licenses often meet (but not necessarily) the requirements of the Open Source Definition [10], used by the Open Source Initiative (OSI) [9] to determine whether a software license can be considered open-source. Such definition does not match exactly that of free software issued by the FSF. However, differences between both terms are very small, so in practice they basically refer to the same software licenses, with a few minor exceptions. According to FSF [6]: *"However, the differences in extension of the category are small: nearly all free software is open source, and nearly all open source software is free."*

Another related term is *shareware*. By it we refer to proprietary software that is provided to users for free on a trial basis only. Usually, shareware is freely provided for a limited period of time in order to offer potential customers the opportunity to use the program thus determining its usefulness and potential advantages over competitors before taking a decision about buying a license. Often shareware is also labeled as "free trial" or "trial version" to stress that meaning. In other cases, the software is offered with some limitations on functionality, such as import/export or save options and the like. Such missing functionalities can be fully activated after purchasing a license. For instance, in videogames, shareware has been traditionally used as a mean to distribute games developed by small-sized companies that do not have access to major distribution channels for their products. In general, shareware games are different than game demos in the sense that the former are much less limited in terms of playtime and number of levels. It is not uncommon that shareware games include the full game, while additional content, assets, extras and other functionalities are only provided with the commercial license.

2.2 Software licenses

As pointed out in previous section, software can be classified according to the rights granted to end-users: availability of software at no cost, access to the source code, possibility of modifying it and distributing it and so on. All those rights are specified in the corresponding software licenses, which can be categorized in a nutshell as proprietary licenses and non-proprietary licenses. In proprietary licenses, the software publisher grants a license to use one or several copies of the software but maintains the ownership of such software so as all rights are retained by the software publisher unless otherwise specified. As a consequence, the user must necessarily accept all terms of the license in order to use the software. Typically, those terms include an extensive list of activities that users are not allowed to perform with the software, such as reverse engineering, any modification of the software, its distribution to third parties and many others.

On the contrary, in non-proprietary licenses the ownership of the copy of the software does not longer remain with the software publisher, but the end-user (different than the ownership of the copyright, that still belongs to the software publisher). As a consequence, end-user can actually use the software without accepting the license. Such acceptance is optional but it is required if end-user also wants to access to other rights such as the modification or distribution rights. At its turn, non-proprietary licenses can be essentially of two different types: copylefted or permissive. The main difference between them is that copylefted licenses state that when modified versions of such software are distributed, they must be distributed under the same terms as the original software. In particular, all improvements or modifications to copylefted software must also be distributed as free software. In other words, modified versions of software with copyleft come also with copyleft. This is sometimes referred to as "share and share alike" or "quid pro quo". The ultimate goal of copyleft is to preserve the freedom and openness of the software itself. A typical example of copyleft is the GNU General Public License [11] (GPL onwards). Under it, end-users can redistribute, reverse engineer, or otherwise modify the software, but all any modifications made and redistributed must include the source code, and the end-user is not allowed to modify the copylefted license for such new software. Additionally, if GPL code is used but not shared or sold, the code is not required to be made available and any changes may remain private. This permits developers and organizations to use and modify GPL code for private purposes without being required to make their changes available to the public.

In opposition, permissive licenses aim to give users total freedom on that software, so users can do basically anything they want with the source code, including the right to use it as a part of software released under a proprietary license. For instance, GPL requires any derivative work that is distributed to be released according to the GPL while permissive licenses, such as Berkeley Software Distribution (BSD), MIT License or University of Illinois/NCSA Open Source License do not; they only require to acknowledge the original authors. As such, permissive licenses are more free than the copylefted ones. In fact, code licensed under a permissive free software license (BSD for instance) can be incorporated into copylefted projects (for example, GPL). The new code emanating from this process becomes GPL compatible. However, the opposite is not true: GPL licensed code cannot be distributed under the BSD license without the previous consent from copyright holders. To summarize, copyleft and permissive licenses are compatible, but their combination can only be distributed under the terms of the copyleft license, not the

permissive license.

A halfway between copyleft and permissive licenses is given by the GNU Lesser General Public license (LGPL) [12]. Under LGPL, copyleft restrictions apply to the program itself but they do not to other software that merely links with the program. This license is mostly used for software libraries, since LGPL libraries can be used by a non-LGPL licensed program (in fact, even by a non-GPL licensed program). A famous example of LPGL application is *OpenOffice*.

A very interesting situation is that of multiple licenses, where software is distributed under two or more different sets of terms and conditions so that end-users can choose which terms they want to use in order to further distribute the software. A typical example are some Mozilla products (such as *Firefox* web browser and *Thunderbird* e-mail client), which are actually tri-licensed, since in addition to the LGPL license, they are also GPL and MPL licensed. That MPL (Mozilla Public License) is a weak copyleft version (source code copied or changed under the MPL must stay under the MPL, but code under the MPL may be combined with proprietary files in one program) approved as a open-source license and a free software license by OSI and FSF respectively.

3 Some Freeware Mathematical Tools

This section provides a comprehensive (although not exhaustive) collection of open-source and freeware mathematical tools and middleware available to handle different tasks regarding the interplay between mathematics and education. Table 1 summarizes all this information. The table is exclusively devoted to non-proprietary software; therefore, the readers will notice that commercial CAS are not reported here. The table lists, in rows, the different computer programs along with the corresponding website, the platforms on which the program is available and, finally, the type of tasks the program is intended for. All these items are reported in columns for each individual program.

It was the intention of the author to report the license information of the different computer packages included in this section. However, this plan collided with the difficulty of the task. Most program licenses have varied greatly over the time, making them difficult to track. Other programs have multiple licenses, depending on several factors, such as the owners, platforms, developer teams and so on. At a certain point, it became clear that providing this information might be useless, since it could potentially become obsolete at the time of reading. So, instead, the reader is kindly advised to check the corresponding website for an up-to-date information regarding the legal issues of the program license and the different options available.

4 Conclusions and Further Remarks

The main conclusion we can derive from the phenomenon of freeware mathematical software is the wide availability of powerful tools to accomplish virtually any task in a way that compares well with commercial software in almost every respect. But, as important as this aspect is, there is also another consideration that deserves full attention. Through the evolution of mathematical software, a repetitive topic has been the confrontation between (apparently) opposite ends. This problem has appeared recurrently

Table 1: Freeware Mathematical Tools

Program	*website*	*Platforms Available*	*Type*
Archim	*www.stochastic-lab.com*	Windows	Graphing tool
Axiom	*axiom-developer.org*	Windows Linux Mac	General-purpose CAS
C.A.R.	*zirkel.sourceforge.net*	Multiplatform (Java program)	Dynamic geometry
CoCoa	*cocoa.dima.unige.it*	Windows Linux Mac UNIX	CAS
EulerMathToolbox	*euler.rene-grothmann.de*	Windows	Numerical tool
FreeMat	*freemat.sourceforge.net*	Windows Linux Mac	Numerical tool
GeoGebra	*www.geogebra.org/cms*	Multiplatform (Java program)	Dynamic geometry
GAP	*www.gap-system.org*	Windows UNIX Mac	Group Theory
Graph	*http://www.padowan.dk*	Windows	Graphing tool
GraphCalc	*www.graphcalc.com*	Linux Windows	Graphing tool
Graph Calculator 3D	*calculator.runiter.com/ math-calculator/*	Windows Linux Mac Unix	Scientific Calculator
KANT/KASH	*page.math.tu-berlin.de/ ~kant*	Windows Linux Mac	Algebraic number theory
Macaulay2	*/www.math.uiuc.edu/ Macaulay2/*	Windows Linux Mac	Algebraic geometry
Mathomatic	*www.mathomatic.org/ math/*	Windows Linux Mac Unix	CAS
MathTrax	*prime.jsc.nasa.gov/ mathtrax/*	Mac Windows	Graphing tool

Table 1 *(cont'd)*

Program	*website*	*Platforms Available*	*Type*
Maxima	*maxima.sourceforge.net*	Windows Linux Mac Unix	CAS
NonEuclid	*www.cs.unm.edu/~joel/NonEuclid/NonEuclid.html*	Multiplatform (Java program)	Dynamic geometry
Octave	*www.gnu.org/software/octave*	Windows Linux Mac Unix	Numerical Tool
PARI-GP	*pari.math.u-bordeaux.fr/*	Windows Linux Mac	CAS
PLURAL	*www.singular.uni-kl.de/plural*	Windows Linux Mac Unix	Non-commutative algebra
Qalculate	*qalculate.sourceforge.net/*	Linux	Scientific calculator
Reduce	*www.reduce-algebra.com*	Windows Linux Mac Unix	General-purpose CAS
SAGE	*www.sagemath.org*	Windows Linux Mac Unix	General-purpose mathematical software
SINGULAR	*www.singular.uni-kl.de/*	Windows Linux Mac Unix	Polynomial computations
Scilab	*www.scilab.org*	Windows Linux Mac	Numerical tool
Winmat	*math.exeter.edu/rparris/winmat.html*	Windows	Linear algebra
Winplot	*math.exeter.edu/rparris/winplot.html*	Windows	Graphing tool

Table 1 *(cont'd)*

Program	*website*	*Platforms Available*	*Type*
WinStat	*math.exeter.edu/ rparris/winstats.html*	Windows	Statistical tool
Wiris	*www.wiris.com/en/cas*	Windows Linux Mac	General-purpose CAS
Giac/Xcas	*www-fourier.ujf-grenoble.fr/ ~parisse/giac.html*	Windows Linux Mac	General-purpose CAS
YACAS	*yacas.sourceforge.net/*	Windows Linux Mac	General-purpose CAS

in computer science (just think about the illustrative examples of Microsoft's Windows vs. Apple's Mac OS X, purists vs. free-spirit coders, or proprietary vs. open source programs) and the mathematical software is certainly not an exception. Remarkable, well-documented examples are the typical "scientific fights" between *Mathematica* and *Maple* followers, between users of numerical and symbolic tools, or between freeware and commercial software.

In opinion of the author this way of thinking (the requirement to choose just one among a myriad of alternative options, much like in a *"with me or against me"* approach) reflects our natural tendency as human beings to grouping, but it is not very intelligent after all. All previous experience shows us that the best and most effective approach to an efficient use of mathematical software in education is to load all this stuff in our backpack and move on. *Why not opening our minds and take the best out of all scientific programs in a cooperative and complementary way?*

Acknowledgements

This paper is the printed version of an invited talk delivered by the author at RIMS (Research Institute for Mathematical Sciences) workshop during the *RIMS Workshop on Mathematical Software and Education: Study on Effective Use of Mathematical Software*, Kyoto University (Japan), on August 22nd. 2012. The author would like to thank the organizers of this exciting RIMS workshop for their diligent work and kind invitation. Special thanks are owed to Prof. Nakamura (Nagoya University) for his patience and support regarding the writing of this paper.

This research has been kindly supported by the Computer Science National Program of the Spanish Ministry of Economy and Competitiveness, Project Ref. #TIN2012-30768, Toho University (Funabashi, Japan), and the University of Cantabria (Santander, Spain).

References

[1] Iglesias, A., Gálvez, A.: Effective BD-binding edutainment approach for powering students' engagement at University through videogames and VR technology. In: International Conference on Convergence Information Technology-ICCIT'2008 - Busan (Korea). *IEEE Computer Society Press*, (2008) 307-314.

[2] Iglesias, A., Ipanaqué, R.: Using computer algebra systems to achieve Bologna's Declaration educational goals. A case study: symbolic proof of limits of functions. *International Journal of Computer Science and Software Technology*, **2**(1) (2009) 35-42.

[3] Iglesias, A.: Facing the challenges of the new European Space of Higher Education through effective use of computer algebra systems as an educational tool. *RIMS Kokyuroku Journal Series*, **1624** (2009) 114-128.

[4] Iglesias, A.: Computer Technologies for XXI Century Education: A New Way to Communicate and Learn at the University of Cantabria. *RIMS Kokyuroku Journal Series*, **1674** (2010) 53-67.

[5] Iglesias, A.: Combining Functional Equations and Computer Algebra Systems with Regard to XXI Century Mathematics Education. *RIMS Kokyuroku Journal Series*, **1735** (2011) 213-223.

[6] The Free Software Foundation: *http://www.gnu.org/philosophy/categories.html*

[7] The Free Software Foundation Web site: *http://www.fsf.org/*

[8] The Free Software Definition: *http://www.gnu.org/philosophy/free-sw.html*

[9] The Open Source Initiative Web site: *http://opensource.org/*

[10] The Open Source Definition: *http://opensource.org/docs/osd*

[11] The GNU General Public License v3.0: *http://www.gnu.org/licenses/gpl-3.0.txt*

[12] The GNU Lesser General Public License: *http://www.gnu.org/copyleft/lesser.html*